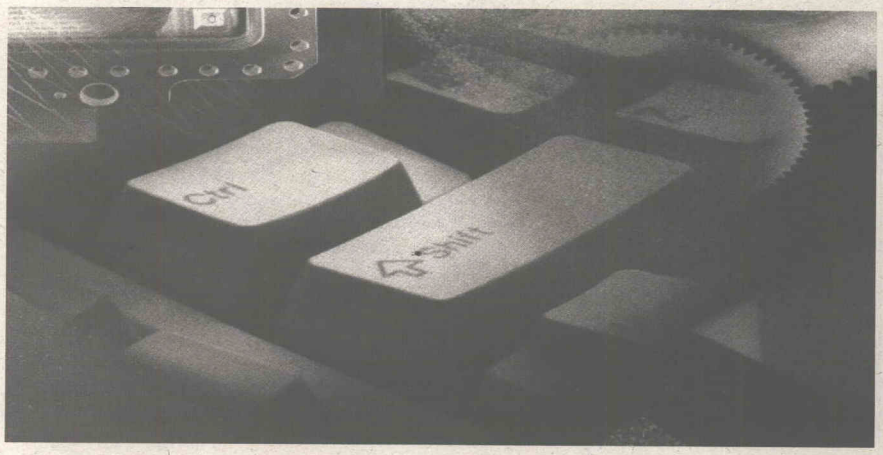
■ สุทธิศักดิ์ พงศ์ธนาพาณิช



Advanced BASIC Programming Technique

การสร้าง ActiveX DLL ตอนที่ 1

คุณสมบัติใหม่ ๆ ของ Visual BASIC 5.0 และ 6.0 ที่เกี่ยวกับ ActiveX

จากตอนที่แล้วของบทความนี้ผู้เขียนได้อธิบายถึงเทคนิคการออปติไมซ์โปรแกรมที่สร้างด้วย Visual BASIC 5.0 (32 บิต) ซึ่งเป็นบทความที่ช่วยเสริมเนื้อหาให้กับหนังสือ Visual BASIC 5.0 Professional : การใช้คำสั่งและคอนโทรล ActiveX ซึ่งแต่งโดยตัวผู้เขียนเอง เป็นหนังสือที่อธิบายถึงคำสั่งและคอนโทรล ActiveX ทั้งหมดของ Visual BASIC 5.0 (และครอบคลุมเนื้อหาที่สามารถนำไปใช้กับ Visual BASIC 6.0 อย่างสมบูรณ์) สำหรับผู้อ่านที่ต้องการศึกษา Visual BASIC 5.0 หรือ 6.0 ตั้งแต่เริ่มต้นจนถึงขั้นที่สามารถใช้คำสั่งและคอนโทรล ActiveX ทั้งหมด สามารถศึกษาได้จากหนังสือเล่มนี้เพียงเล่มเดียวเท่านั้น และหากท่านใดสงสัยเกี่ยวกับเนื้อหาหรือ Visual BASIC สามารถสอบถามได้ โดยส่งมาตามแอดเดรสอีเมล์ sutthisakp@yahoo.com ผู้เขียนจะได้อธิบายให้ทราบคร่าว ๆ เป็นราย ๆ ไปครับ

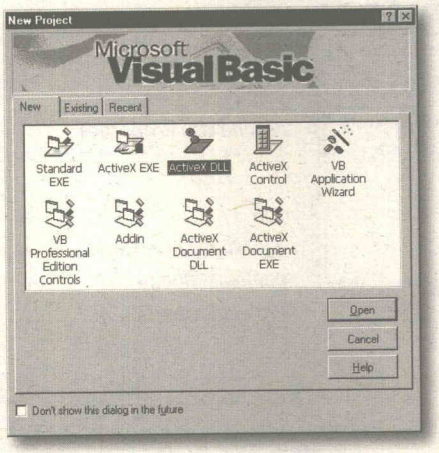
สำหรับในบทความตอนนี้ผู้เขียนก็จะอธิบายถึงคุณสมบัติใหม่ ๆ ของ Visual BASIC 5.0 และ 6.0 ที่เกี่ยวกับ ActiveX ซึ่งได้ถูกกำหนดโดยไมโครซอฟท์ให้เป็นสถาปัตยกรรมพื้นฐานของวินโดวส์ 32 บิต ในปัจจุบัน โดยจะเริ่มต้นจากการแนะนำวิธีการสร้าง แก้ไข และทดสอบองค์ประกอบ ActiveX DLL (สำหรับผู้อ่านที่ใช้ Visual BASIC 4.0–32 บิต ไมโครซอฟท์จะเรียกองค์ประกอบ ActiveX DLL ว่าองค์ประกอบ in-process) ซึ่งองค์ประกอบ ActiveX DLL จะช่วยให้การพัฒนาแอพพลิเคชันเป็นไปอย่างรวดเร็ว และสามารถลดความซ้ำซ้อนของโค้ดได้มากทีเดียว และนอกจากนี้ยังสามารถที่จะแบ่งโค้ดใช้งานรวมกับแอพพลิเคชันอื่น ๆ ได้อีกด้วย

รู้จักกับองค์ประกอบ ActiveX DLL

องค์ประกอบ ActiveX หมายถึง หน่วยหรือชุดของโค้ดคำสั่ง (executable code) ที่ถูกคอมไพล์ให้อยู่ในรูปแบบของไฟล์ .exe, .dll หรือ .ocx โดยที่ไฟล์ดังกล่าวจะต้องเป็นไปตามข้อกำหนดมาตรฐาน ActiveX ของไมโครซอฟท์ ซึ่งเทคโนโลยี ActiveX นี้จะช่วยให้โปรแกรมเมอร์สามารถที่จะรวมเอาองค์ประกอบเหล่านี้ เพื่อ

นำกลับมาใช้ใหม่ในการสร้างซอฟต์แวร์หรือแอพพลิเคชันต่าง ๆ โดยไม่จำเป็นต้องเขียนหรือสร้างโค้ดขึ้นมาใหม่ ซึ่งจะช่วยลดเวลาในการพัฒนาซอฟต์แวร์ และง่ายต่อการแก้ไขหรือดัดแปลง และนอกจากนี้สำหรับการพัฒนาเพื่อใช้ในระดับองค์กร โปรแกรมเมอร์ยังสามารถที่จะพัฒนาองค์ประกอบ ActiveX จนมีความเป็นมาตรฐานสำหรับการเรียกใช้งานได้อีกด้วย ซึ่งผู้เขียนจะได้กล่าวรายละเอียดต่อไป

การพัฒนาซอฟต์แวร์โดยอาศัยองค์ประกอบ ActiveX จะแตกต่างจากการโปรแกรมเชิงวัตถุ (OOP, Object-Oriented Programing) โดยที่การโปรแกรมเชิงวัตถุเป็นวิธีหรือรูปแบบสำหรับการสร้างองค์ประกอบของซอฟต์แวร์ในรูปแบบพื้นฐานเชิงวัตถุ (object-based) แต่ ActiveX เป็นเทคโนโลยีหรือมาตรฐานที่ช่วยให้องค์ประกอบในรูปแบบของพื้นฐานเชิงวัตถุสามารถที่จะถูกนำมารวมกันหรือใช้งานร่วมกันได้ (OOP เน้นเรื่องการสร้างออบเจ็กต์ และ ActiveX เน้นในเรื่องการนำออบเจ็กต์มาใช้งานร่วมกันหรือให้สามารถทำงานร่วมกันได้)



รูปที่ 1 แสดงไดอะล็อกบ็อกซ์ New Project

เมื่อผู้อ่านมีความเข้าใจถึงความหมายและวัตถุประสงค์ของการสร้างองค์ประกอบ ActiveX พอสมควรแล้ว ต่อไปผู้เขียนก็จะอธิบายลึกลงไปเฉพาะในส่วนของ ActiveX DLL ซึ่งเป็นเนื้อหาของบทความตอนนี้ โดยจะเริ่มต้นที่ ActiveX DLL หมายความว่าอย่างไร? คำตอบก็คือ ActiveX DLL ในความเป็นจริงก็คือไฟล์ .dll ชนิดหนึ่งที่มีมาตรฐานข้อกำหนดตามเทคโนโลยี ActiveX ซึ่งเป็นไฟล์ .dll ที่ไม่สามารถทำงานได้โดยลำพัง (ต่างจากไฟล์ .dll โดยทั่วไปที่สร้างโดย Visual C++ ที่เป็นไฟล์ .dll แท้ ที่สามารถทำงานได้โดยลำพังและสามารถทำงานได้เร็วกว่า ActiveX DLL) และในการเรียกใช้โค้ดในองค์ประกอบ ActiveX DLL ก็ต้องใช้วิธีการสร้างออบเจ็กต์ (โดยใช้ประโยค Dim และ Set ดังที่จะได้กล่าวต่อไป) เพื่อลิงก์ไปยังคลาสในองค์ประกอบ ActiveX DLL ส่วนการทำงานองค์ประกอบ ActiveX DLL จะทำงานในขบวนการ (process) อื่น ๆ ของแอพพลิเคชันที่เรียกใช้งาน

การสร้างโปรเจ็กต์สำหรับ ActiveX DLL

จากอีเมล์ที่มีเข้ามาผู้เขียนพอที่จะทราบว่า ผู้อ่านบทความนี้ต้องการที่จะให้เพิ่มรายละเอียดของการอธิบายและให้อธิบายแบบทีละขั้นทีละตอน หรือจับมือเขียนโค้ดกันเลย ดังนั้นในบทความตอนนี้ผู้เขียนก็จะขออนุญาตเปลี่ยนแนวการเขียนบทความก็จะพยายามอธิบายอย่างละเอียด ถ้าหากผู้อ่านท่านใดมีข้อคิดเห็นประการใดเกี่ยวกับวิธีการเขียนบทความแบบนี้ ก็ช่วยอีเมล์มาคุยกันบ้างครับ

สมมติว่าในขณะนี้ผู้อ่านได้เปิดเอนไวรอนเมนต์ของ Visual Basic 5.0 ที่ยังคงว่างอยู่ในปัจจุบัน ขอให้ผู้อ่านทำตามขั้นตอนต่อไปนี้ เพื่อเริ่มต้นสร้างโปรเจ็กต์สำหรับ ActiveX DLL ที่ชื่อ DemoActiveX DLL

1. เลือกคำสั่ง New Project ของเมนู File (เมนูซ้ายมือสุด) เพื่อสร้างโปรเจ็กต์ใหม่

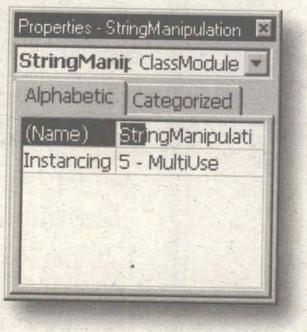
2. จากนั้นจะปรากฏไดอะล็อก-บ็อกซ์ New Project ดังในรูปที่ 1 เพื่อให้ผู้อ่านได้เลือกประเภทของโปรเจ็กต์ที่ต้องการสร้างโดย ในที่นี้ขอให้ผู้อ่านเลือกที่ไอคอน ActiveX DLL โดยการดับเบิลคลิกที่ไอคอนดังกล่าว หรือเลือกไอคอนดังกล่าวแล้วคลิกที่ปุ่มคำสั่ง Open จากนั้น Visual Basic ก็จะสร้างโปรเจ็กต์ชื่อ Project1 ซึ่งประกอบด้วย 1 โมดูลคลาส ชื่อ Class1 สำหรับเป็นพิมพ์เขียวในการสร้างออบเจ็กต์

3. เลือกคำสั่ง Properties Window ในเมนู View หรือโดยการกดคีย์ F4 ก็จะปรากฏหน้าต่าง Properties (ในกรณีที่หน้าต่างเป็นแบบ dockable หน้าต่างก็อาจจะเรียงลำดับอยู่ที่ด้านข้างของเอนไวรอนเมนต์แล้วก็ได้) จากนั้นก็เลือกที่ช่อง Name (คุณสมบัติ Name) เพื่อแก้ไขชื่อของโมดูลคลาสเป็น StringManipulation (ในทางปฏิบัติผู้อ่านสามารถตั้งชื่อเป็นอย่างอื่นก็ได้ครับ เช่น Nirvana เป็นต้น) โดยที่ชื่อของโมดูลคลาสนี้จะถูกนำไปใช้ในการสร้างออบเจ็กต์ต่อไป

หมายเหตุ สำหรับคุณสมบัติ Instancing ที่เหลือในหน้าต่าง Properties จะใช้ในการกำหนดให้แอพพลิเคชันปลายทางสามารถที่จะสร้างหลาย ๆ ตัวอย่าง (instance) ของออบเจ็กต์จากโมดูลคลาสนี้หรือไม่ ซึ่งผู้เขียนจะไม่ขอกล่าวถึงในที่นี้ และขอให้ผู้อ่านเลือกที่ค่า 5-MultiUse ไปก่อนก็แล้วกันครับ

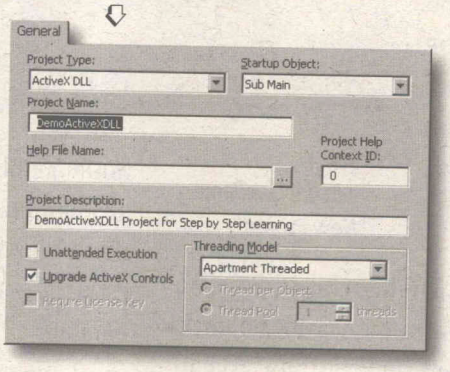
Advanced BASIC

1. เลือกคำสั่ง Project1 Properties จากเมนู Project ก็จะปรากฏไดอะล็อกบ็อกซ์ Project Properties เพื่อให้ผู้อ่านได้กำหนดคุณสมบัติต่าง ๆ ของโปรเจ็กต์ (ถ้าหากโปรเจ็กต์ที่ถูกเปิดในขณะนั้นไม่ได้ชื่อ Project1 คำสั่งนี้ก็จะถูกเปลี่ยนชื่อเป็นคำสั่งที่ขึ้นต้นด้วยชื่อใหม่ของโปรเจ็กต์โดยอัตโนมัติทันที) โดยที่ภายในไดอะล็อกบ็อกซ์ Project Properties จะประกอบด้วย 4 แถบป้าย (tab) เพื่อแยกคุณสมบัติต่าง ๆ ของโปรเจ็กต์ตามวัตถุประสงค์การใช้งาน เพื่อให้ง่ายต่อการเลือกแก้ไขคุณสมบัติต่าง ๆ ของโปรเจ็กต์ โดยที่แต่ละแถบป้ายจะประกอบด้วยคำสั่งที่มีความหมายดังต่อไปนี้



รูปที่ 2
แสดงหน้าต่าง Properties

รูปที่ 3
แสดงแถบป้าย General ของไดอะล็อกบ็อกซ์ Project Properties



แถบป้าย General

ประกอบด้วย รายการสำหรับกำหนดคุณสมบัติของโปรเจ็กต์ ดังนี้

- Project Type กำหนดชนิดของโปรเจ็กต์ เช่น ActiveX DLL เป็นต้น ซึ่งถ้าผู้อ่านมีการเปลี่ยนแปลงชนิดของโปรเจ็กต์ คุณสมบัติอื่น ๆ ที่เกี่ยวข้องก็จะถูกเปลี่ยนไปด้วยโดยอัตโนมัติ
- Startup Object กำหนดฟอร์มหรือรูทีน Sub Main ของโมดูลมาตรฐาน สำหรับใช้เริ่มต้นรันโปรเจ็กต์
- Project Name กำหนดชื่อของโปรเจ็กต์ ซึ่งจะเป็นชื่อที่ถูกใช้ในการลงทะเบียนของวินโดวส์ โดยที่ชื่อของโปรเจ็กต์นี้จะปรากฏในไดอะล็อกบ็อกซ์ Object Browser ถ้าหากมีการอ้างอิงถึงโดยการเขียนโค้ด
- Help File Name กำหนดชื่อของไฟล์ help (.hlp) ซึ่งผู้อ่านสามารถเลือกได้จากไดอะล็อกบ็อกซ์โดยการคลิกที่ปุ่มคำสั่ง ...
- Project Help Context ID กำหนดหมายเลข context ID ของหัวข้อ Help ที่เกี่ยวข้อง
- Project Description กำหนดคำอธิบายโปรเจ็กต์ ซึ่งคำอธิบายนี้จะปรากฏที่ด้านล่างของไดอะล็อกบ็อกซ์ Object Browser แต่ถ้าหากผู้อ่านทำการสร้าง ActiveX DLL หรือ ActiveX EXE คำอธิบายนี้ก็จะปรากฏที่ด้านล่างของไดอะล็อกบ็อกซ์ References
- Upgrade ActiveX Controls กำหนดให้มีการอัพเกรดคอนโทรล ActiveX จากเวอร์ชันที่ผ่านมาเป็นคอนโทรล ActiveX เวอร์ชันล่าสุดที่มากับ Visual BASIC 5.0 หรือ 6.0
- Require License Key กำหนดให้โปรเจ็กต์ ActiveX ที่ถูกสร้างต้องการรหัสอนุญาตการใช้งาน (License Key) ซึ่ง Visual BASIC จะทำการสร้างไฟล์ license (.vbl) เมื่อมีการสร้างโปรเจ็กต์ขึ้นมา
- Unattended Execution กำหนดให้โปรเจ็กต์ทำงานโดยไม่มีการตอบสนองต่อผู้ใช้ ซึ่งโปรเจ็กต์ชนิด Unattend จะไม่มีอินเทอร์เฟส ซึ่งผู้อ่านสามารถเลือกได้ 2 ชนิด ดังนี้

1. Thread per Object

กำหนดให้แต่ละ instance ของคลาสเป็นแบบชนิด Multiuse ซึ่งจะถูกสร้างภายใต้ส่วนของงาน (thread) ที่แตกต่างกัน ซึ่งจะส่งผลให้แต่ละส่วนของงาน จะมีข้อมูลของตัวแปรชนิดครอบคลุมเป็นของตัวเองอย่างอิสระ

2. Thread Pool

กำหนดให้แต่ละ instance ของคลาสเป็นแบบชนิด Multiuse ซึ่งจะถูกสร้างภายใต้ส่วนของงานจาก thread pool โดยจะส่งผลให้แต่ละส่วนของงานจะมีข้อมูลของตัวแปรชนิดครอบคลุมชุดเดียวกัน ซึ่งในกรณีนี้ผู้อ่านจำเป็นต้องกำหนดจำนวนของส่วนของงานที่จะถูกสร้างสำหรับ thread pool โดยการเลือกตัวเลขจากรายการ Number of threads

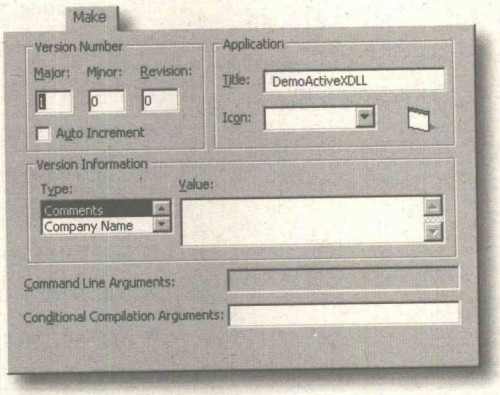
สำหรับในตัวอย่างนี้ เพื่อความง่ายก็ขอให้ผู้อ่านเลือกแก้ไขเฉพาะรายการต่อไปนี้

- เลือกรายการใน Project Type เป็น Active .DLL
- เลือกรายการใน Startup Object เป็น Sub Main
- แก้ไขชื่อในช่องของ Project Name เป็น DemoActiveXDLL
- แก้ไขชื่อในช่อง Project Description เป็น DemoActiveXDLL Project Step by Step Learning

● และเลือกตัวเลือกของรายการ Upgrade ActiveX Controls

หมายเหตุ เมื่อผู้อ่านเสร็จสิ้นขั้นตอนนี้ในข้อ 4 ผู้เขียนขอให้ผู้อ่านคลิกที่ปุ่มคำสั่ง OK แล้วทดลองเลือกคำสั่งเดียวกันจากเมนู Project ก็จะเห็นว่าชื่อของคำสั่งเปลี่ยนไปเป็น DemoActiveXDLL Properties ทันทีโดยอัตโนมัติ

สำหรับคำอธิบายต่อไปนี้ ผู้เขียนจะขออธิบายเพื่อให้ผู้อ่านที่สนใจศึกษาเพิ่มเติม ได้ทราบถึงความหมายของคำสั่งต่าง ๆ ในแถบป้ายที่เหลือ เพื่อให้สามารถปรับแต่งโปรเจ็กต์ได้ตามที่ต้องการ แต่สำหรับการสร้าง ActiveX DLL ตามตัวอย่างนี้ ผู้อ่านสามารถกระโดดข้ามไปยังข้อที่ 5 ได้ทันทีครับ



รูปที่ 4 แสดงแถบป้าย Make ของไดอะล็อกบ็อกซ์ Project Properties

แถบป้าย Make

ประกอบด้วย รายการสำหรับกำหนดคุณสมบัติของโปรเจ็กต์ สำหรับการสร้างไฟล์ .exe เช่น หมายเลขเวอร์ชัน ไอคอน หรือรายละเอียดอื่น ๆ โดยที่แถบพื้นที่แสดงผล Make นี้จะเทียบเท่ากับแถบป้ายชื่อเดียวกันของคำสั่ง Make Project ของเมนู File ทุกประการ

● Version Number กำหนดหมายเลขเวอร์ชันของไฟล์ไบนารี

– Major หมายเลขเวอร์ชันหลัก ซึ่งสามารถมีค่าได้ตั้งแต่ 0 ถึง 9,999

– Minor หมายเลขเวอร์ชันรอง ซึ่งสามารถมีค่าได้ตั้งแต่ 0 ถึง 9,999

– Revision หมายเลขเวอร์ชันการแก้ไข ซึ่งสามารถมีค่าได้ตั้งแต่ 0 ถึง 9,999

– Auto Increment ถ้าหากถูกเลือก Visual BASIC ก็จะทำการเพิ่มหมายเลขการแก้ไข (Revision) ครั้งละ 1 ทุกครั้งที่มีการสร้างไฟล์ไบนารี

● Version Information กำหนดข้อมูลรายละเอียดของไฟล์ไบนารีเวอร์ชันปัจจุบัน ดังนี้

– Value สำหรับกรอกข้อมูลของรายการต่อไปนี้

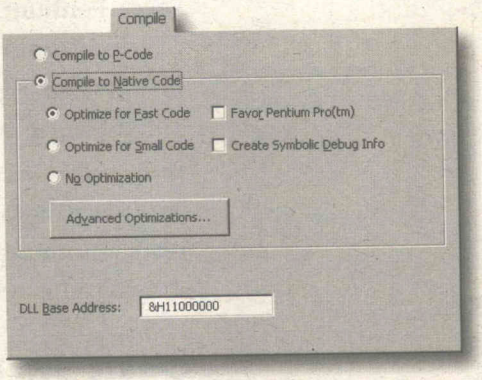
1. Comments ข้อมูลหมายเหตุเพื่ออธิบายแอพพลิเคชัน
2. Company Name ชื่อบริษัทหรือองค์กร
3. File Description ข้อมูลรายละเอียดอธิบายไฟล์ที่ถูกสร้าง
4. Legal Copyright ข้อมูลการสงวนลิขสิทธิ์
5. Legal Trademarks ข้อมูลการจดทะเบียนเครื่องหมายการค้า
6. Product Name ข้อมูลเกี่ยวกับผลิตภัณฑ์

– Application กำหนดชื่อ และไอคอนของแอพพลิเคชัน ดังนี้

1. Title กำหนดชื่อของแอพพลิเคชัน
2. Icon กำหนดไอคอนสำหรับแสดงผลแอพพลิเคชัน

– Command Line Arguments กำหนดค่าสำหรับใช้เป็น Command Line ให้กับแอพพลิเคชันในการทดสอบการทำงานด้วยคำสั่ง Run ของ Visual Basic

– Conditional Compilation Arguments กำหนดค่าคงที่ สำหรับใช้ในการคอมไพล์อย่างมีเงื่อนไข ซึ่งผู้อ่านสามารถใช้ฟังก์ชัน #If...#Then...#Else ในการทดสอบค่าคงที่ดังกล่าวได้



รูปที่ 5 แสดงแถบป้าย Compile ของไดอะล็อกบ็อกซ์ Project Properties

แถบป้าย Compile

ประกอบด้วย รายการสำหรับการกำหนดเงื่อนไขสำหรับการสร้างไฟล์ .exe เช่น การสร้างไฟล์ .exe ในรูปแบบของโค้ดเทียม (p-code) หรือโค้ดแท้ (native code) ซึ่งผู้อ่านสามารถที่จะเลือกได้ว่าจะทำการออปติไมซ์ได้ทั้งขนาดของไฟล์หรือความรวดเร็วในการทำงาน โดยที่แถบป้าย Compile นี้จะเทียบเท่ากับแถบป้ายชื่อเดียวกันของคำสั่ง Make Project ของเมนู File ทุกประการ

- Compile to P-Code สร้างโค้ดสำหรับไฟล์ไบนารีด้วยโค้ดเทียม
- Compile to Native Code สร้างโค้ดสำหรับไฟล์ไบนารีด้วยโค้ดแท้ ซึ่งผู้อ่านสามารถทำการออปติไมซ์โค้ดแท้ ได้หลาย ๆ วิธีดังนี้

Optimize for Fast Code

กำหนดให้คอมไพเลอร์สร้างโค้ดเพื่อให้ไฟล์ไบนารีที่ได้ทำงานได้รวดเร็วที่สุด โดยในขั้นตอนการสร้างโค้ดแท้นั้น คอมไพเลอร์จะพยายามที่จะสร้างชุดคำสั่งที่ทำงานได้รวดเร็วที่สุดเท่าที่จะเป็นไปได้ ซึ่งในบางครั้งขนาดของไฟล์ไบนารีที่ได้อาจจะมีขนาดโตกว่าปกติก็ได้

Optimize for Small Code

กำหนดให้คอมไพเลอร์สร้างโค้ดเพื่อให้ได้ไฟล์ไบนารีที่มีขนาดเล็กที่สุด โดยในขั้นตอนการสร้างโค้ดแท้นั้น คอมไพเลอร์จะพยายามที่จะสร้างชุดคำสั่งที่มีขนาดกะทัดรัดที่สุดเท่าที่จะเป็นไปได้ ซึ่งผลลัพธ์ที่ได้ส่วนใหญ่จะทำให้ไฟล์ไบนารีที่ได้มีความเร็วในการทำงานลดลง

No Optimization

ไม่มีการออปติไมซ์โค้ดไบนารี

Favor Pentium Pro(tm)

สร้างโค้ดไบนารีที่เหมาะสำหรับการทำงานบนเครื่องที่ใช้โปรเซสเซอร์ Pentium Pro (P6) ของบริษัทอินเทล ซึ่งโค้ดที่ได้จะสามารถทำงานบนเครื่องที่ใช้โปรเซสเซอร์ Pentium Pro ได้อย่างมีประสิทธิภาพมากที่สุด แต่ถึงอย่างไรก็ตาม โค้ดไบนารีที่ได้ก็ยังคงสามารถทำงานบนโปรเซสเซอร์ก่อนหน้านี้ได้โดยไม่มีปัญหา เพียงแต่ประสิทธิภาพในการทำงานลดลงเท่านั้น

Create Symbolic Debug Info

กำหนดให้มีการสร้างข้อมูลสำหรับ symbolic debug ลงในไฟล์ไบนารี ซึ่งจะทำให้ผู้อ่านสามารถทำการดีบักไฟล์ไบนารีที่ได้โดยใช้ Visual C++ 5.0 โดยที่คอมไพเลอร์จะทำการสร้างไฟล์สำหรับจัดเก็บข้อมูลสำหรับ symbolic debug โดยจัดเก็บลงในไฟล์ที่มีนามสกุล .pdb

Advanced Optimizations

แสดงผลไดอะล็อกบ็อกซ์ Advanced Optimizations เพื่อให้ผู้อ่านที่มีความรู้เกี่ยวกับการเขียนโค้ดสำหรับ Visual Basic เป็นอย่างดี ได้ทำการออปติไมซ์ในส่วนของการตรวจสอบข้อผิดพลาดบางรายการ ทั้งนี้เพื่อช่วยเพิ่มความรวดเร็วในการทำงานของแอพพลิเคชันที่ได้ ซึ่งมีหลายรายการให้เลือกกำหนด (โปรดศึกษารายละเอียดการกำหนดและเงื่อนไขเพิ่มเติมได้จากหนังสือ Visual BASIC 5.0 Professional : การใช้คำสั่งและคอนโทรล ActiveX)

แถบป้าย Component

ประกอบด้วย รายการสำหรับการกำหนดโหมดสำหรับการเริ่มต้นรันโปรเจ็กต์ และควบคุมความเข้ากันได้ของคอนโทรลที่ต่างเวอร์ชัน (Version compatibility) ซึ่งรายการต่าง ๆ ที่ผู้อ่านสามารถกำหนดให้กับคอนโทรล ActiveX ดังต่อไปนี้

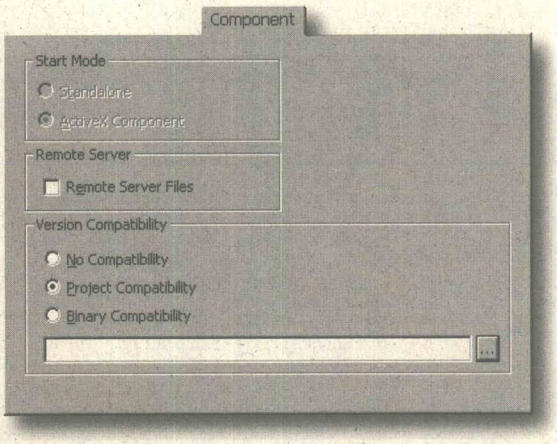
- **Start Mode** กำหนดโหมดการทำงานของโปรเจ็กต์ ดังนี้

1. Standalone ทำงานแบบแอพพลิเคชันทั่วไป
2. ActiveX Component ทำงานแบบองค์ประกอบของ ActiveX

- **Remote Server** กำหนดให้มีการใช้คุณสมบัติ DCOM ในการติดต่อกับไฟล์ข้ามเครื่องคอมพิวเตอร์โดยอาศัยระบบเครือข่าย โดยถ้าหากผู้อ่านต้องการใช้ความสามารถของคุณสมบัติดังกล่าว ก็ให้เลือกให้ปรากฏเครื่องหมาย / หน้าคำสั่ง Remote Server Files
- **Version compatibility** กำหนดความเข้ากันได้ของคอนโทรล ActiveX ที่ถูกสร้างขึ้นมาหรือแก้ไขใหม่ ซึ่งมีอยู่ด้วยกัน 3 วิธี ดังนี้

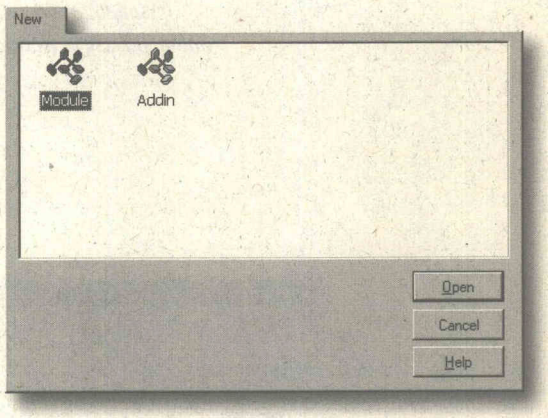
1. No Compatibility ไม่สามารถเข้ากันได้กับคอนโทรล ActiveX ที่ถูกสร้างขึ้นมาก่อนหน้านี้
2. Project Compatibility กำหนดให้สามารถเข้ากันได้กับคอนโทรล ActiveX ที่ถูกสร้างขึ้นมาก่อนหน้านี้ ซึ่งถ้าหากผู้อ่านไม่มีการกำหนดค่าของ Version compatibility โดยปกติมันก็จะถูกกำหนดให้เป็น Project Compatibility
3. Binary Compatibility กำหนดให้สามารถเข้ากันได้กับคอนโทรล ActiveX ที่ถูกสร้างขึ้นมาก่อนหน้านี้ในระดับของโค้ดไบนารี ซึ่งจะสามารถทำให้ใช้งานได้กับทุก ๆ แอพพลิเคชันที่มีการเรียกใช้คอนโทรล ActiveX ที่ถูกสร้างขึ้นมาก่อนหน้านี้

5. เลือกคำสั่ง Add Module ในเมนู Project ซึ่งจะปรากฏไดอะล็อกบ็อกซ์ Add Module เพื่อให้ผู้อ่านได้เลือกประเภทของโมดูล โดยในรูปที่ 7 จะปรากฏเป็นไอคอนให้เลือก เป็นโมดูลมาตรฐาน และโมดูล Add-In ตามลำดับ สำหรับในที่นี้ก็ขอให้ผู้อ่านดับเบิลคลิกที่ไอคอน Module ของโมดูลมาตรฐาน



รูปที่ 6
แสดงแถบป้าย Component
ของไดอะล็อกบ็อกซ์ Project Properties

รูปที่ 7
แสดงไดอะล็อกบ็อกซ์ Add Module



6. จากนั้นก็ให้ผู้อ่านลอกโค้ดดังต่อไปนี้ลงในส่วนประกาศของ (declaration) โมดูลมาตรฐาน (Module1) โดยให้ผู้อ่านคีย์โค้ดต่อไปนี้ลงในหน้าต่างโค้ด (หน้าต่าง Code) ใต้โค้ดบรรทัด Option Explicit ลงมา โดยให้ผู้อ่านกดคีย์ Enter ทุกครั้งที่สิ้นสุดแต่ละบรรทัด

```
Option Explicit

' internal-DLL-used variable for CreatedDate property.
Public glbCreatedDate As Variant

Sub Main()
 ' keep current module starting date.
 glbCreatedDate = Now
 ' display the message in Debug window.
Debug.Print "Main procedure executed!!!"
End Sub
```

7. ขั้นตอนสุดท้ายในการสร้างโปรเจ็กต์ DemoActiveX DLL ก็คือ การจัดเก็บทุก ๆ ไฟล์ของโปรเจ็กต์ลงในดิสก์ สำหรับโปรเจ็กต์ตัวอย่างนี้ ผู้อ่านต้องทำการจัดเก็บทั้งหมด 3 ไฟล์ ดังต่อไปนี้

1. ไฟล์โมดูลมาตรฐาน (module) ชื่อ DemoActive XDLL_Module.bas

2. ไฟล์โมดูลคลาส (class module) ชื่อ DemoActive XDLL_StringManipulation.cls

3. ไฟล์โปรเจ็กต์ (project) ชื่อ DemoActive XDLL.vbp

สำหรับขั้นตอนข้างต้นทั้ง 7 ขั้นตอน ถือได้ว่าเป็นขั้นตอนมาตรฐานเริ่มต้นในการสร้างโปรเจ็กต์ สำหรับการสร้างองค์ประกอบ ActiveX DLL เพื่อให้ง่ายต่อการจดจำผู้เขียนก็จะขอทบทวนเป็นขั้นตอนสั้น ๆ ดังต่อไปนี้

1. สร้างโปรเจ็กต์ใหม่

2. เลือกประเภทของโปรเจ็กต์ใหม่ที่จะสร้างให้เป็นโปรเจ็กต์ชนิด ActiveX DLL

3. เปิดไดอะล็อกบ็อกซ์ Properties เพื่อกำหนดคุณสมบัติ Name ของโปรเจ็กต์ชนิด ActiveX DLL

4. เปิดไดอะล็อกบ็อกซ์ Project Properties เพื่อกำหนดคุณสมบัติหลาย ๆ ประการ ที่เกี่ยวกับโปรเจ็กต์ชนิด ActiveX DLL ซึ่งจะถูกนำไปใช้ในการสร้างไฟล์ .dll ต่อไป

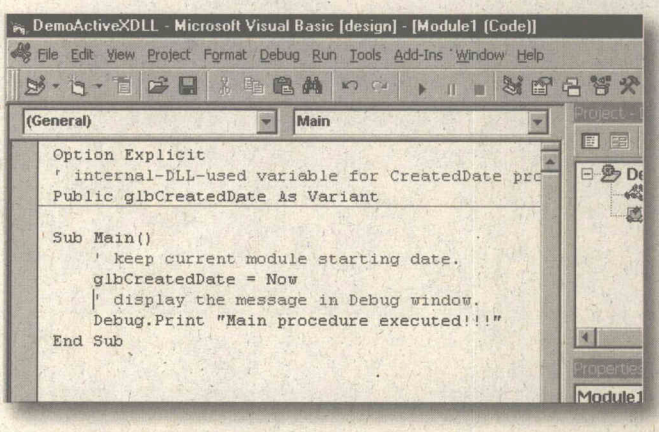
5. เพิ่มโมดูลมาตรฐานให้กับโปรเจ็กต์ โดยการเลือกคำสั่ง Add Module

6. เขียนโค้ดลงในโมดูลมาตรฐาน (.bas)

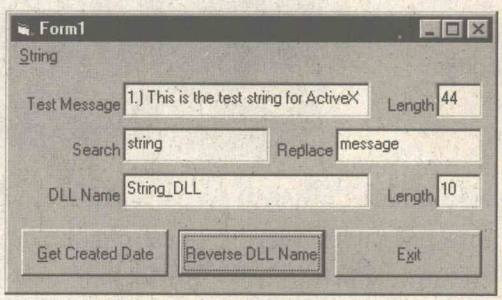
7. จัดเก็บโมดูลมาตรฐาน (.bas) โมดูลคลาส (.cls) และโปรเจ็กต์ (.vbp) ลงบนดิสก์

ซึ่งหลังจากที่เสร็จสิ้นขั้นตอนข้างต้นแล้ว ต่อไปผู้อ่านต้องเริ่มต้นสร้างออบเจ็กต์ โดยการกรอกโค้ดที่จำเป็นรวมทั้งการแก้ไขโค้ดที่เกี่ยวข้องกับคุณสมบัติ (Properties) หรือวิธี (Methods) ให้กับออบเจ็กต์ในโมดูลคลาส (.cls) เพื่อให้ออบเจ็กต์มีความสมบูรณ์และเหมาะที่จะนำไปใช้งานมากขึ้น จากนั้นก็จะเป็นขั้นตอนการเขียนโค้ดเพื่อกำหนดค่าเริ่มต้น (initialization) และกำหนดการสิ้นสุด (termination) ให้กับการทำงานของออบเจ็กต์ที่ถูกสร้างจากโมดูลคลาส ซึ่งในทางปฏิบัติ Visual BASIC ได้เตรียมโพรซีเดอร์เหตุการณ์ (event procedure) สำหรับขบวนการทั้งสองให้อยู่แล้วโดยอัตโนมัติ ซึ่งในส่วนที่เหลือก็จะเป็นเพียงขั้นตอนการเขียนโค้ด ตามความต้องการของโปรแกรมเมอร์เท่านั้น

Advanced BASIC

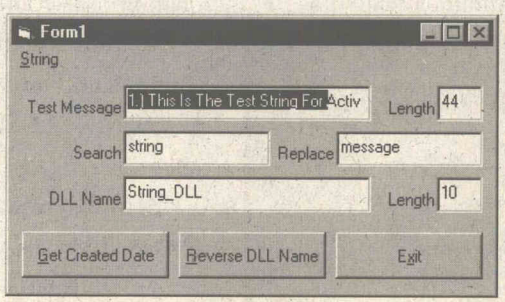


รูปที่ 8
แสดงรูปโค้ด
ที่กรอกลงในหน้าต่าง Code
ของโมดูลมาตรฐาน (Module1)



รูปที่ 9
แสดงฟอร์มของโปรแกรม
TestDemoActiveXDLL.exe

รูปที่ 10
แสดงผลลัพธ์
ของการคลิกที่ปุ่มคำสั่ง
Get Created Date



รูปที่ 11
แสดงผลลัพธ์ของการเลือกคำสั่ง
Capital Word

โดยในรูปที่ 10 ถึง 11 ได้แสดงให้เห็นถึงผลลัพธ์ของการคลิกที่ปุ่มคำสั่ง Get Created Date และการเลือกคำสั่ง Capital Word เพื่อแสดงถึงวันที่และเวลาที่ ActiveX DLL ได้ถูกสร้างในหน่วยความจำ และการแปลงข้อความที่ปรากฏในช่อง Test Message ให้ตัวอักษรตัวแรกของทุก ๆ คำ (word) เป็นตัวอักษรพิมพ์ใหญ่ทั้งหมดตามลำดับ สำหรับรายละเอียดการทำงานและการสร้างไฟล์ทั้งสอง ผู้อ่านก็จะได้ทราบอย่างละเอียดในบทความตอนต่อ ๆ ไปครับ สวัสดีครับ

ข้อมูลเสริมพิเศษ

: ข้อมูลเชิงเทคนิคสำหรับ Visual BASIC 5.0

ก่อนที่จะจบบทความผู้เขียนก็จะขออธิบายถึงข้อจำกัดทั้งหมดของ Visual Basic 5.0 มีทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ เพื่อช่วยให้ผู้อ่านสามารถออกแบบซอฟต์แวร์ ที่เขียนโดย Visual Basic ได้อย่างถูกต้อง ดังต่อไปนี้

ด้านฮาร์ดแวร์

- ระบบปฏิบัติการวินโดวส์ 95, 98, NT3.51+
- ไมโครโปรเซสเซอร์ 80486 หรือสูงกว่า (แนะนำ Pentium 90+)
- หน่วยความจำ RAM 8 เมกะไบต์ขึ้นไป (แนะนำ 16 เมกะไบต์ สำหรับ 95, 98 และ 32 เมกะไบต์ สำหรับ NT3.51+)
- จอภาพแสดงผล VGA

และขั้นตอนสุดท้ายก็จะเป็นการสร้างหรือคอมไพล์โปรเจ็กต์ให้เป็นไฟล์ .dll เพื่อให้สามารถนำไปใช้งานได้ทันที ซึ่งรายละเอียดทั้งหมดตามที่กล่าวมาข้างต้น ผู้เขียนจะขอยกยอดไปพูดคุยกันต่อในคราวหน้าครับ แต่ก่อนจบบทความตอนนี้ผู้เขียนก็จะแนะนำวิธีการใช้โปรแกรมตัวอย่างเพื่อทดสอบ ActiveX DLL ที่สร้างขึ้นมาเป็นตัวอย่างของบทความนี้ (ไฟล์ดังกล่าวยังไม่มีในซอฟต์เมล์สำหรับบทความตอนนี้ครับ) โดยเมื่อผู้อ่านรันไฟล์ ซึ่งจะปรากฏฟอร์มของโปรแกรม โดยในรูปที่ 9 ได้แสดงถึงฟอร์มในขณะเริ่มต้น โดยให้ผู้อ่านกรอกข้อความตามที่ปรากฏลงในช่อง Test Message ช่อง Search และช่อง Replace จากนั้นก็ให้ทดลองใช้โปรแกรมโดยการคลิกที่ปุ่มคำสั่งต่าง ๆ หรือเลือกคำสั่งจากเมนู String ก็ได้

Advanced BASIC

คุณสมบัติ	คอนโทรล	ข้อจำกัด
List	ListBox, ComboBox	จำนวนรายการไม่เกิน 32 KB
Text	TextBox	จำนวนตัวอักษรไม่เกิน 64 KB
Caption	Label	จำนวนตัวอักษรไม่เกิน 1 KB
	CommandButton, CheckBox	จำนวนตัวอักษรไม่เกิน 255 ตัวอักษร
	Frame, OptionButton	จำนวนตัวอักษรไม่เกิน 32 KB
Tag	ทุกคอนโทรล	จำกัดโดยขนาดของหน่วยความจำ
Name	ทุกคอนโทรล	จำนวนตัวอักษรไม่เกิน 40 ตัวอักษร

ด้านซอฟต์แวร์

- โปรเจ็กต์

– ชื่อของตัวแปร จะต้องมีความยาวของจำนวนตัวอักษรน้อยกว่า 255 ตัวอักษร

– การตั้งชื่อของฟอร์ม (form) คอนโทรล (control) โมดูลมาตรฐาน (module) และคลาสโมดูล (class) จะต้องมีความยาวของจำนวนตัวอักษรน้อยกว่า 40 ตัวอักษร

– ไม่มีข้อจำกัดด้านจำนวนของออบเจ็กต์ที่แตกต่างกันในแต่ละโปรเจ็กต์

- คอนโทรล

– คอนโทรลทั้งหมดที่ไม่ใช่คอนโทรลกราฟิก (ไม่ใช่คอนโทรล Shape, Line, Image และ Label) จะถูกสร้างในรูปแบบหน้าต่าง โดยที่แต่ละหน้าต่างจะใช้ทรัพยากรของระบบ ซึ่งจะเป็นตัวกำหนดจำนวนของหน้าต่างที่จะสามารถมองเห็นได้หรือแสดงผลในเวลาเดียวกัน

– จำนวนของชื่อของคอนโทรลในแต่ละฟอร์ม จะต้องไม่เกิน 254 ชื่อของคอนโทรล

– คอนโทรลอะเรย์จะมีค่าสมาชิกของอะเรย์ได้ตั้งแต่ 0 ถึง 32,767 เท่านั้น

– การวางคอนโทรลซ้อนลงในพื้นที่แสดงผลของคอนโทรลอื่น ๆ ในรูปแบบของการจัดชั้น (layer) เช่น การวางคอนโทรล Frame ลงในคอนโทรล Frame อื่น ๆ จะทำได้ไม่เกิน 6 ชั้นเป็นอย่างมาก

– สำหรับบางคอนโทรล จะมีข้อจำกัดปลีกย่อย ดังตารางที่ 1

- โค้ดของโปรแกรม

– สำหรับแต่ละฟอร์ม โมดูลมาตรฐานหรือคลาสโมดูล สามารถมีโค้ดได้มากที่สุด 65,534 บรรทัด

– โค้ดแต่ละบรรทัด จะสามารถมีความยาวได้มากที่สุด 1,023 ไบต์

– การสร้างความต่อเนื่องของโค้ดแต่ละบรรทัดด้วยเครื่องหมาย _ (underscore) สามารถกระทำได้มากที่สุดโดยที่มีเครื่องหมาย _ มากที่สุดไม่เกิน 25 เครื่องหมาย

– จำนวนโค้ด (จำนวนตัวอักษรทั้งหมดรวมเครื่องหมาย) ในแต่ละโพรซีเดอร์ ต้องไม่เกิน 64 KB

– จำนวนโพรซีเดอร์ในแต่ละโมดูล จะไม่มีการจำกัดจำนวน

– จำนวนประโยคในการประกาศฟังก์ชัน DLL ทั้งหมดในแต่ละฟอร์มหรือโมดูล จะต้องไม่เกิน 1,500 การประกาศ (โดยประมาณ)

- ข้อมูล

– สำหรับแต่ละฟอร์ม โมดูลมาตรฐานหรือคลาสโมดูล สามารถมีเซกเมนต์ข้อมูล (data segment) ได้โตที่สุด 64 KB

– ข้อมูลประเภทสร้างโดยผู้ใช้ (user-defined type) มีขนาดโตที่สุด 64 KB

– สแต็ก (stack) จะถูกใช้งานโดยพารามิเตอร์ต่าง ๆ เมื่อมีการรันโปรแกรมเท่านั้น

เทคนิคประจำคอลัมน์

ถาม ขอทราบวิธีในการสร้างตัวเลขสุ่มด้วย Visual BASIC?

ตอบ ตัวเลขสุ่ม (random number) หมายถึง ตัวเลขที่ถูกสร้างขึ้นมาด้วยวิธีการสุ่ม ดังนั้นจึงไม่สามารถที่จะคาดเดา ตัวเลขสุ่มตัวถัดไปได้อย่างถูกต้อง 100 เปอร์เซ็นต์ ซึ่งในการสร้างตัวเลขสุ่มโดยปกติเราจะใช้ฟังก์ชัน Rnd ซึ่งจะทำหน้าที่สร้างตัวเลขสุ่มมีค่าตั้งแต่ 0 ถึง 1 ขึ้นมา

โดยในโพรซีเดอร์ RandomNumber จะทำหน้าที่ตัวเลขสุ่มชนิดจำนวนเต็มมีค่าตั้งแต่ Lower ถึง Upper ขึ้นมา โดยมีโค้ดดังต่อไปนี้

```
Public Function RandomNumber (Upper As Long,_
                    Lower As Long) As Long
        Randomize
        RandomNumber = Clng ((Upper - Lower + 1)_
                * Rnd + Lower)
End Function
```

สำหรับประโยค Randomize จะทำหน้าที่กำหนดให้การสร้างตัวเลขสุ่มโดยฟังก์ชัน Rnd ในแต่ละครั้งจะใช้ตัวเลขไม่ซ้ำชุดเดิม